REMARKS

This Amendment A is responsive to the first Office Action mailed August 11, 2003. Applicants respectfully submit that claims 1, 3-9, 11-13, and 15-18 as set forth herein patentably distinguish over the cited references, and ask for allowance of these claims.

The current status of the claims

Claims 1-18 stand rejected under 35 U.S.C. §102(e) as being anticipated by Ponnekanti et al. (U.S. 6,363,387, hereinafter "Ponnekanti").

Claim 18 stands rejected under 35 U.S.C. §112, first paragraph as failing to comply with the enablement requirement.

Claims 17-18 stand rejected under 35 U.S.C. §112, second paragraph as being indefinite.

The 35 U.S.C. § 112 rejections have been addressed

Claim 18 has been amended to replace "and" with "or" as suggested by the Examiner the Office Action. **Claims 17-18** have been amended to specify that the unique key index table is a unique key index table managed by the index manager. This change was also suggested by the Examiner in the Office Action.

In view of these amendments, applicants respectfully ask for withdrawal of the 35 U.S.C. § 112 rejections of **claims 17-18.**

THE PRESENT APPLICATION

The present application addresses and overcomes certain deadlock situations created by concurrent transactions in a database having a unique key index. A deadlock condition may be created when, for example, two

substantially simultaneous delete transactions involving different rows are followed up by two substantially simultaneous insert transactions that attempt to insert row data into the rows having the row identifications (RID's) of the deleted rows.

In database systems utilizing pseudo-deletion of index entries, a deadlock situation can arise if the two simultaneous insert transactions have index key values corresponding to the deleted rows. In this case, each insert transaction may acquire an X-lock for the pseudo-deleted index entry matching the other insert transaction's key value to enter data into that row. Each insert transaction will also attempt to acquire an S-lock on the pseudo-deleted row having index key value corresponding to the index key value of the inserted data. This S-lock acquisition attempt is performed to verify that the pseudo-deleted row is actually deleted (that is, the delete has committed), so that the inserted row will have a unique key value. Each S-lock becomes blocked by the X-lock of the other insert transaction, thus causing deadlock.

The deadlock occurs because the system is unable to recognize that the blocking X-locks are not associated with the delete transactions, which have already committed. The present application overcomes this deadlock by providing:

(1) a new X-lock attribute, called a "Delete" attribute, that is set when an X-lock is granted to a Delete transaction; and

(2) a new Conditional S-lock, that is:

a. not compatible with an X-lock with the lock Delete attribute set;

b. compatible with an X-lock with the Delete attribute NOT set.

Specification at page 20 line 19 through page 21 line 4.

The delete attribute element (1) is an attribute of a lock, and serves to distinguish the X-lock acquired by a delete operation from X-locks acquired by other types of operations. In FIGURES 4(B) and 4(C), the Delete element **450** appearing in the Lock Table is the new X-lock delete attribute. The delete X-lock attribute (that is, element (1) above) is different from the pseudo-delete flag, such as the pseudo-delete flag **412** shown in FIGURES 4(B) and 4(C).

The X-lock delete attribute is distinguished from the pseudo-delete flag attribute of an index entry, which appears as the symbol "D" in the Index Table. See at least at page 16 lines 7-12. The delete X-lock attribute disappears when the X-lock is removed after the delete is committed; in contrast, the pseudo-delete flag **412** remains set even after the delete is committed.

The new Conditional S-lock is conditional with respect to the delete X-lock attribute, and is granted when there is no X-lock or when there is an X-lock but that lock does not correspond to a delete operation (that is, the delete X-lock attribute is OFF). Thus, granting of the Conditional S-lock ensures that either there is no X-lock or, if there is an X-lock, then that X-lock is not associated with a delete operation.

THE PONNEKANTI REFERENCE

Ponnekanti does not disclose a delete X-lock attribute.

Ponnekanti does disclose a delete bit for index entries: "Each index row employs a ROW_DELETE bit." Ponnekanti col. 12 lines 58-64 (underscore added). This index ROW_DELETE bit corresponds to the pseudo-delete index entry attribute disclosed in the present application. The

index ROW DELETE bit cannot simultaneously correspond to the X-lock delete attribute.

Ponnekanti also discloses a data ROWDELETE bit. Ponnekanti col. 12 lines 34-37. The ROWDELETE status bit is associated with the row, and not with the X-lock obtained by the delete operation. This distinction is shown at col. 12 lines 36-57, which explain that a delete operation "sets the ROW_DELETE bit but the contents of the data row are left intact." Col. 12 lines 40-42. The ROW_DELETE status bit is reset when a subsequent transaction locks onto the row, at which time the row delete bit can be removed. Thus, the ROWDELETE status bit remains after the delete operation's X-lock is removed - it therefore cannot be associated the X-lock of the delete operation.

This is further shown at Ponnekanti col. 13 lines 31-51, which describes the use of a "lock instant." The "lock instant" appears to correspond to the unconditional S-lock of the present application, and is used "to see whether there exists a conflicting lock already held on the row." Col. 13 lines 36-38. When a "delete" status bit is found, the "lock instant " is requested. If no conflicting lock is found, then the "lock instant" is granted and the operation issuing the "lock instant" knows that the delete operation has been committed. Col. 13 lines 38-40. This further demonstrates that the delete bit is not a lock attribute - it may be that the delete bit is set even though there is no lock on the row.

On the other hand, if a conflicting lock does exist, the "lock instant" fails, and is queued (i.e., sleeps) until the conflicting lock is removed. Col. 13 lines 46-48. This is the normal operation of an unconditional S-lock when a conflicting X-lock is already applied. There is no disclosure or fair suggestion to

provide a lock attribute of the X-lock to identify whether the conflicting lock was issued by a delete operation. Without such an X-lock attribute, the system cannot tell whether an X-lock conflicting with the unconditional S-lock is due to a delete operation or some subsequent operation, such as another insert operation. Two insert operations attempting substantially simultaneous S-locks can thus be deadlocked, and this deadlock is not overcome by Ponnekanti.

As noted by the Office Action, Ponnekanti also uses the expression "conditional lock request." However, this conditional lock request is entirely different from the Conditional lock request disclosed in the present application.

Ponnekanti defines an unconditional lock request and a conditional lock request. In the case of an unconditional lock request that requests a lock on an already locked row, the unconditional lock request is queued (i.e., sleeps on the lock) until the other transaction releases its lock. Ponnekanti col. 11 lines 36-41. In the case of a conditional lock request that requests a lock on an already locked row "the Lock Manager returns LOCK_DIDNTQUEUE status and does not grant the lock." Ponnekanti col. 11 lines 42-44.

The "conditional" aspect of Ponnekanti's conditional lock request resides in whether or not the lock is granted. If a conflicting lock exists, the conditional lock is not granted. In contrast, Ponnekanti's unconditional lock request is always granted, although it may be queued until a conflicting lock releases. In contrast, the Conditional lock of the present application is conditioned on whether a conflicting X-lock has its Delete attribute OFF.

To summarize, Ponnekanti does not disclose the Delete X-lock attribute of the present application, which identifies an X-lock as associated with a delete operation. Ponnekanti also does not disclose the Conditional S-lock of the present application that is granted if a conflicting X-lock has its X-lock delete attribute not set. As a result, Ponnekanti does not resolve the deadlock situation illustrated, for example, in FIGURES 3(A)-3(F) of the present application, because the conditional S-lock of Ponnekanti cannot distinguish between an X-lock of a delete operation and an X-lock of another operation.

Claims 1-6 patentably distinguish over the cited references

Claim 1 has been amended to call for the exclusive X-lock to include a Delete X-lock attribute associated therewith, a SET state of the Delete X-lock attribute being indicative of a transaction holding the X-lock being a delete transaction. Claim 1 has been further amended to incorporate the subject matter of canceled claim 2 directed toward the Conditional S-lock. Specifically, amended claim 1 calls for:

> an unconditional S-lock that enables shared access to the first table element and is selectively assigned by the lock manager to the first table element only when the first table element is without an exclusive X-lock previously assigned thereto; and
>
> a conditional S-lock that enables shared access to the first table element and is selectively assigned by the lock manager to the first table element only when the first table element is either without an exclusive X-lock previously assigned thereto or is without an

exclusive X-lock having its Delete X-lock attribute SET assigned thereto.

Although Ponnekanti discloses conditional and unconditional S-locks, these elements are wholly different from what is called for in amended claim 1. In Ponnekanti, the "conditional" S-lock is conditional in that if an X-lock exists, the conditional S-lock is simply not granted. There is no disclosure in Ponnekanti of a Delete X-lock attribute, much less of a Conditional S-lock whose granting is conditioned upon either the absence of a conflicting X-lock or the absence of a conflicting X-lock having its Delete X-lock attribute SET.

For at least these reasons, Applicants respectfully submit that **claim 1**, as well as **claims 3-6** that depend therefrom, as set forth herein are in condition for allowance and request an early indication of allowance of these claims.

Claims 7-9 patentably distinguish over the cited references

Claim 7 has been amended to call for requesting a Conditional S-lock on a table row indexed by the pseudo-deleted table index entry, said Conditional S-lock having compatibility characteristics respective to an X-lock:

the Conditional S-lock not including compatible with an X-lock having a Delete attribute that is SET or ON, and

the Conditional S-lock being compatible with an X-lock having a Delete attribute that is NOT SET or OFF.

Ponnekanti does not disclose a conditional S-lock having these characteristics. Rather, what Ponnekanti

calls a conditional S-lock is granted conditional upon whether there is any conflicting X-lock, not on the attributes of the conflicting X-lock.

Claim 8 sets forth that step of receiving an indication that the Conditional S-lock is granted includes the steps of:

granting the Conditional S-lock conditional upon the table row indexed by the pseudo-deleted table index entry not having an X-lock assigned thereto;

granting the Conditional S-lock conditional upon the table row indexed by the pseudo-deleted table index entry having an X-lock assigned thereto wherein said X-lock has its Delete attribute not set, reset, or off; and

receiving an indication that the Conditional S-lock is granted conditional upon the granting of the Conditional S-lock.

Claim 8 specifies that the Conditional S-lock is granted in the case where the X-lock assigned thereto has its Delete attribute not set, reset, or off. In contrast, neither the conditional nor the unconditional S-lock of Ponnekanti is ever granted when an X-lock is also present. In the conditional case, a conflicting X-lock in Ponnekanti simply causes the conditional S-lock to fail, whereas in the unconditional case a conflicting X-lock in Ponnekanti causes the S-lock to be queued (or to "sleep," as Ponnekanti puts it).

For at least these reasons, Applicants respectfully submit that **claims 7-9** as set forth herein are in condition for allowance and request an early indication of allowance of these claims.

Claims 11-13 patentably distinguish over the cited references

Claim 11 has been amended to call for requesting a Conditional S-lock on a table row indexed by the pseudo-deleted table index entry, said Conditional S-lock being incompatible with an X-lock acquired by a delete operation and being compatible with an X-lock not acquired by a delete operation. The conditional S-lock of Ponnekanti does not distinguish between an X-lock acquired by a delete operation and an X-lock not acquired by a delete operation. As a result, under certain circumstances the database of Ponnekanti can experience deadlock if an S-lock applied to verify that a delete operation has committed is blocked by an X-lock applied by an operation other than the Delete operation. In contrast, by using the Conditional S-lock as called for in claim 11 such a deadlock does not arise, because the Conditional S-lock recognizes that the conflicting X-lock was not acquired by a delete operation. Hence, the delete operation must have committed, and so the Conditional S-lock is granted.

Claim 12 further distinguishes over Ponnekanti, which does not disclose or fairly suggest granting the Conditional S-lock conditional upon the table row indexed by the pseudo-deleted table index entry having an X-lock assigned thereto wherein said X-lock has a Delete attribute that is not set, reset, or off, as called for in claim 12.

For at least these reasons, Applicants respectfully submit that **claims 11-13** as set forth herein are in condition for allowance and request an early indication of allowance of these claims.

Claims 15-18 patentably distinguish over the cited references

Claim 15 calls for a lock manager for use in a database management system (DBMS) managing a database application including a database having at least one table and cooperative with an index manager and a data manager for restricting access to a first table element of said at least one table by assigning one or more locks thereto including at least an exclusive X-lock that enables exclusive access to the first table element, the exclusive X-lock including a Delete attribute associated therewith, a SET state of the Delete attribute being indicative of a transaction holding the X-lock being a delete transaction.

Ponnekanti does not disclose an exclusive X-lock including a Delete attribute associated therewith in which a SET state of the Delete attribute is indicative of a transaction holding the X-lock being a delete transaction. By including a delete X-lock attribute as called for in claim 15, a subsequent insert transaction can determine whether or not an X-lock on a pseudo-deleted row means that the delete transaction has not yet committed.

Claim 16 calls for the lock manager to be adapted to restrict access to said first table element by assigning a Conditional S-lock that enables shared access to the first table element and is selectively assigned by the lock manager to the first table element only when the first table element does not have an X-lock with its Delete attribute in a SET state assigned thereto. Ponnekanti does not disclose a Conditional S-lock with the behavior called for in claim 16.

Claim 18 calls for the lock manager to be operative to grant the Conditional S-lock on the table row corresponding to the existing index key entry only when the table row is

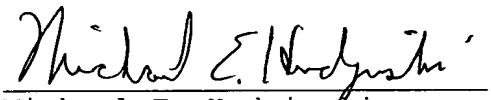
without an exclusive X-lock assigned thereto, or the table row has an exclusive X-lock assigned thereto with its Delete attribute not in said SET state, to enable the index manager to update the table index key entry with the new row identification RID, release the Conditional S-lock on the table row corresponding to the existing index key entry, and reset the pseudo-delete flag to and OFF state. Ponnekanti does not disclose granting a conditional S-lock when the table row has an exclusive X-lock assigned thereto with its Delete attribute not in said SET state. Indeed, Ponnekanti does not disclose a Delete attribute of an exclusive X-lock.

CONCLUSION

For the reasons set forth above, it is submitted that all claims 1, 3-9, 11-13, and 15-18 as set forth herein patentably distinguish over the references of record. Allowance of all claims and an early notice to that effect is respectfully requested.

Respectfully submitted,

FAY, SHARPE, FAGAN,
MINNICH & McKEE, LLP



Michael E. Hudzinski
Reg. No. 34,185
1100 Superior Avenue
Seventh Floor
Cleveland, Ohio 44114-2518
(216) 861-5582

L:\RWS\DATA\IBM20007\IBM007AA.DOC